Scaling Up: Distributed Machine Learning with Cooperation[*]

Foster John Provost
NYNEX Science & Technology
400 Westchester Avenue
White Plains, NY 10604
foster@nynexst.com

Daniel N. Hennessy
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260
hennessy@cs.pitt.edu

Abstract

Machine-learning methods are becoming increasingly popular for automated data analysis. However, standard methods do not scale up to massive scientific and business data sets without expensive hardware. This paper investigates a practical alternative for scaling up: the use of distributed processing to take advantage of the often dormant PCs and workstations available on local networks. Each workstation runs a common rule-learning program on a subset of the data. We first show that for commonly used rule-evaluation criteria, a simple form of cooperation can guarantee that a rule will look good to the set of cooperating learners if and only if it would look good to a single learner operating with the entire data set. We then show how such a system can further capitalize on different perspectives by sharing learned knowledge for significant reduction in search effort. We demonstrate the power of the method by learning from a massive data set taken from the domain of cellular fraud detection. Finally, we provide an overview of other methods for scaling up machine learning.

Introduction

Machine-learning techniques are prime candidates for automated analysis of large business and scientific data sets. Large data sets are necessary for higher accuracy (Catlett, 1991b), for learning small disjuncts with confidence, and to avoid over-fitting with large feature sets. However, the standard tools of the machine-learning researcher, such as off-the-shelf learning programs on workstation platforms, do not scale up to massive data sets. For example, Catlett estimates that ID3 (Quinlan, 1986) would take several months to learn from a million records in the flight data set from NASA (Catlett, 1991a).

One solution to this scaling problem is to invest in or to gain access to very powerful hardware. Another is to design alternative methods that can deal better with massive data sets. In this paper, we investigate a third solution, namely, to take advantage of existing processing power distributed across a local network and often under-utilized. In particular, we focus on partitioning the set of examples and distributing the subsets across a network of heterogeneous workstations. We use a standard rule-learning algorithm, modified slightly to allow cooperation between learners.

At a high level, our metaphor for distributed learning is one of cooperating experts, each of which has a slightly different perspective on the concept to be learned. We define *cooperation* as the learning-time sharing of information to increase the quality of the learned knowledge or to reduce or redirect the search. The learners communicate with each other by passing messages. The group can take advantage of the communication by asking questions or by sharing learned knowledge.

We present a practical method for scaling up to very large data sets that can be guaranteed to learn rules equivalent to those learned by a *monolithic* learner, a learner operating on the entire data set. In the next section we show how to guarantee that every rule that a monolithic learner would judge to be satisfactory would appear to be satisfactory to at least one of our distributed learners. Next we discuss how distributed rule learners can take advantage of this property by cooperating to ensure that the ensemble learns *only* rules that are satisfactory over the entire data set. We present results demonstrating the distributed system's ability to scale up when learning from a massive set of cellular fraud data. Later we show how further cooperation can increase the scaling substantially. Finally we discuss other approaches to scaling up.

Partitioning and Accuracy Estimates

If learning algorithms had access to the probability distribution over the example space, then a useful definition of the quality of a learned rule would be the probability that the class indicated by the rule is correct when its conditions apply to an example. Unfortunately, the probability distribution is not usually available. Thus, statistics from the training set typically are used to estimate the probability that a rule is correct. The *positive predictive value* discussed by Weiss, *et al.* (1990), is a frequency-based accuracy estimate; the rule certainty factor used by Quinlan (1987) is a frequency-based accuracy estimate adjusted for small samples, and several rule-learning programs use the Laplace accuracy estimate (Clark & Boswell, 1991; Segal & Etzioni, 1994; Webb,

[*] Appears in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).*

1995; Quinlan & Cameron-Jones, 1995). We show that a distributed learner can make performance guarantees with respect to each of these rule quality metrics.

It is useful to begin by defining some terms. A rule, *r*, is a class description that will either cover or not cover each example in a data set, *E*. Thus, coverage statistics can be determined for *r* and *E*. Let *P* and *N* be the numbers of positive and negative examples in *E*. The number of true positives, *TP*, and the number of false positives, *FP*, count the positive and negative examples covered by *r*. For a subset, E_i, of E, T_i, N_i, TP_i, and FP_i are defined analogously.

Let us define a *rule evaluation criterion* to be the combination of a *rule evaluation function*, $f(r,E)$, which takes a rule and an example set and produces a scalar evaluation score, and a *threshold*, *c*. With respect to the rule evaluation criterion, a rule, *r*, is *satisfactory* over an example set, *E*, if $f(r,E) \geq c$. Rule evaluation criteria can be defined for each of the three rule quality metrics referenced above by defining the appropriate rule evaluation function. For positive predictive value, $f(r,E) = ppv(r,E) = TP/(TP+FP)$; for the certainty factor used by Quinlan, $f(r,E) = cf(r,E) = (TP-0.5)/(TP+FP)$; for the Laplace accuracy estimate, $f(r,E) = le(r,E) = (TP+1)/(TP+FP+k)$, where *k* is the number of classes in the data.

Given a set of examples, *E*, and a partition of *E* into *N* disjoint subsets, E_i, $i=1..N$, the *invariant-partitioning property* (introduced by Provost & Hennessy (1994)) is the phenomenon that for some rule evaluation criteria the following holds for all partitions of *E*: if a rule *r* is satisfactory over *E*, then there exists an *i* such that *r* is satisfactory over E_i. The implication of the invariant-partitioning property is that distributed learning algorithms can be designed such that each processor has only a subset of the entire set of examples, but every rule that would appear satisfactory to a monolithic learner will appear satisfactory to at least one distributed learner. It is straightforward to show that the invariant-partitioning property holds for positive predictive value.

Unfortunately, it is also straightforward to show that the property does not hold for the rule certainty factor used by Quinlan or for the Laplace accuracy estimate. However, by extending the property to allow weakened criteria on the subsets we can provide the same performance guarantees for these rule evaluation functions.

In particular, given a set of examples, *E*, a partition of *E* into *N* disjoint subsets, E_i, $i=1..N$, and a secondary function $f'(r,E,N)$, define a rule to be *acceptable* over an example subset, E_i, if $f'(r,E_i,N) \geq c$, *i.e.*, the rule is satisfactory with respect to *f'*. The *extended* invariant-partitioning property is the phenomenon that for some rule evaluation criteria the following holds for all partitions of *E*: if a rule *r* is satisfactory over *E*, then there exists an *i* such that *r* is acceptable over E_i. The usefulness of the extended property hinges on the definition of *f'*.

The global performance guarantee with the extended property is the same as with the original, namely, *every rule that is satisfactory to a monolithic learner will be acceptable to at least one distributed learner*. With the original property, a rule was acceptable only if it was satisfactory. A weaker definition of acceptability will allow more rules to be found by the distributed learners. Below we utilize cooperation to ensure that spurious rules are eliminated. We now show that for a non-trivial *f'* the extended property holds for the Laplace accuracy estimate.

Define the *Laplace estimate criterion* as: $f(r,E)=le(r,E)$, $c=L$. Define $f'(r,E,N) = le'(r,E,N) = (TP+1/N)/(TP+FP+k/N)$. As expected, $le'(r,E,N) \cong le(r,E)$, which means that the criterion used on the subsets is approximately the same as that used on the entire data set. In fact, it is easy to verify that for $N=1$, $le'(r,E,N) = le(r,E)$; as $N \rightarrow \infty$, $le'(r,E,N) \rightarrow ppv(r,E)$, and for $N>1$, $le'(r,E,N)$ is between $le(r,E)$ and $ppv(r,E)$.

Assume that for a rule, *r*: $(TP+1)/(TP+FP+k) \geq L$, but, given a partition of *N* subsets of *E*: $\forall i, (TP_i +1/N)/ (TP_i + FP_i+k/N) < L$ (i.e., *r* is not acceptable over any E_i), then:

1) $\forall i \{TP_i + 1/N < L \cdot (TP_i + FP_i + k/N)\}$
2) $\Sigma(TP_i +1/N) < \Sigma(L \cdot (TP_i+FP_i + k/N))$
3) $\Sigma(TP_i +1/N) < L \cdot \Sigma(TP_i+FP_i + k/N)$
4) $TP + 1 < L \cdot (TP + FP + k)$
5) $(TP+1)/(TP+FP+k) < L$ ==> Contradiction

Furthermore, it can be shown that *le'* is tight; it is the strongest function for which the extended invariant-partitioning property will hold. By using a similar derivation, it is easy to show that the extended property applies to the certainty factor used by Quinlan. It also applies to the certainty factor normalized for skewed distributions. Specifically, $f(r,E) = cf\_normalized(r,E) = (TP-0.5)/(TP+\rho FP)$, where ρ is the ratio of positive examples to negative examples in the training set.

Cooperating Distributed Learners

We have designed and implemented DRL (Distributed Rule Learner) taking advantage of the invariant-partitioning property. DRL partitions and distributes the examples across a network of conventional workstations each running an instance of a rule learning program. In DRL the learners cooperate based on the communication of partial results to each other. The invariant-partitioning property guarantees that any rule that is satisfactory on the entire data set will be found by one of the sub-learners. Simple cooperation assures that *only* rules that are satisfactory on the entire data set will be found. Later we will discuss more elaborate cooperation.

DRL

DRL is based upon RL (Clearwater & Provost, 1990). RL performs a general-to-specific beam search of a syntactically defined space of rules, similar to that of other MetaDENDRAL-style rule learners (Buchanan & Mitchell, 1978; Segal & Etzioni, 1994; Webb 1995), for rules that satisfy a user-defined rule evaluation criterion. For this work, we use *cf_normalized* (defined above).

DRL first partitions the training data into *N* disjoint subsets, assigns each subset to a machine, and provides the infrastructure for communication when individual learners detect an acceptable rule. When a rule meets the evaluation criterion for a subset of the data, it becomes a *candidate* for meeting the evaluation criterion globally; the extended invariant-partitioning property guarantees that each rule that is satisfactory over the entire data set will be acceptable over at least one subset. As a local copy of RL discovers an acceptable rule, it broadcasts the rule to the other machines to review its statistics over the rest of the examples. If the rule meets the evaluation criterion globally, it is posted as a satisfactory rule. Otherwise, its local statistics are replaced with the global statistics and the rule is made available to be further specialized. Initially, the review of acceptable rules has been implemented as an additional process that examines the entire data set.

Empirical Demonstration

We have been using a rule-learning program to discover potential indicators of fraudulent cellular telephone calling behavior. The training data are examples of cellular telephone calls, each described by 31 attributes, some numeric, some discrete with hundreds or thousands of possible values. The data set used for the experiments reported here comprises over 1,000,000 examples. High-probability indicators are used to generate subscriber behavior profilers for fraud detection. We chose a set of parameters that had been used in previous learning work on the fraud data for monolithic RL as well as for DRL.

The invariant-partitioning property is observed. In order to examine whether the invariant-partitioning property is indeed observed (as the above theory predicts), we examined the rules learned by the multiple processors in runs of DRL using multiple processes on multiple workstations (as described below) and compared them to the rules learned by a monolithic RL using the union of the DRL processors' data sets. As expected, the individual DRL processes learned different rule sets: some did not find all the rules found by the monolithic RL; some produced spurious rules that were not validated by the global review. However, as predicted by the invariant-partitioning property, the final rule set produced by DRL was essentially the same as the final rule set produced by the monolithic RL. The only difference in the rule sets was that DRL found some extra, globally satisfactory rules not found by RL. This is due to the fact that RL conducts a heuristic (beam) search. Because of the distribution of examples across the subsets of the partition, some processors found rules that had fallen off the beam in the monolithic search. Thus, the distributed version actually learned *more* satisfactory rules than the monolithic version in addition to learning substantially faster.

Scaling up. Figure 1 shows the run times of several different versions of the rule learner as the number of examples increases: monolithic RL (RL), a semi-serial version of DRL, and DRL running on four processors plus a fifth for the rule review. RL's run time increases linearly in the number of examples, until the example set no longer fits in main memory, at which point the learner thrashes, constantly paging the example set during matching. It is possible to create a serial version of DRL that operates on the subsets one after the other on a single machine, in order to avoid memory-management problems when the example sets become large. However, because it does not exhibit true (learning-time) cooperation, there is a significant overhead involved with the further specialization of locally acceptable rules that are not globally satisfactory, which is necessary to guarantee performance equivalent to monolithic RL. *Semi-serial-DRL* uses a second processor for the rule review, thus avoiding much of the aforementioned overhead. Figure 1 also includes a line corresponding to five times the run time of DRL (DRL*5) to illustrate the efficiency of the distribution.

For a fixed number of examples, the run time for each DRL processor does not change significantly as the number of processors increases, suggesting that communication overhead is negligible. For the DRL system used in this demonstration, thrashing set in at just over 300,000 examples (as expected).

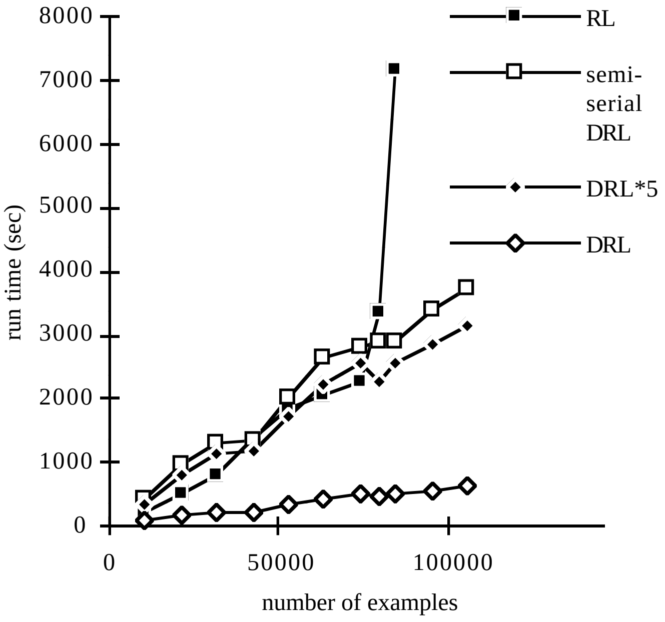

Figure 1. Run time vs. number of examples for the fraud data. (averages over 3 runs). DRL uses 4 workstations + 1 for rule review.

We are interested in the real-time expense of using such systems, so these are real-time results, generated on a university laboratory network of DECstation 5000's with 32M of main memory. Since the major non-linearity hinges on the amount of main memory, we also experimented with dedicated Sparc10's with 64M of main memory. For RL's run time, the shape of the graph is the same. Runs with 100,000 examples per processor take approximately 20 minutes on the Sparc10s; thrashing sets in just under 300,000 examples. This implies that with 5 machines available, DRL can process a million examples while you go get lunch.

The semi-serial version of DRL provides a practical method for dealing with very large example sets even when many processors are not available. The invariant-partitioning property allows it to make the same performance guarantees as DRL; the partitioning makes it very efficient by avoiding the scaling problems associated with memory management.

Further cooperation

The study described above uses a simple form of cooperation to provide a guarantee of an output equivalent to that of a monolithic learner operating with the entire data set. In this section we discuss three further ways in which cooperation can be used to benefit a set of distributed rule learners. Specifically, we discuss sharing learned knowledge for (i) maximizing an accuracy estimate, (ii) pruning portions of the rule space that are guaranteed not to contain satisfactory rules, and (iii) pruning portions of the rule space heuristically.

The invariant-partitioning property is based on learning rules whose evaluation function is greater than a threshold. Some existing rule learners search for rules that maximize the positive predictive value (Weiss, *et al.*, 1990) or the Laplace estimate (Webb, 1995; Segal & Etzioni, 1994). DRL can approximate the maximization process by starting with a high threshold and iteratively decreasing the threshold if no rules are found. However, a system of distributed learners can take advantage of cooperation to maximize the rule evaluation function directly. Specifically, each learner keeps track of the score of the globally best rule so far (initially zero). When a learner finds a rule whose local evaluation exceeds the threshold defined by the global best, it sends the rule out for global review. The invariant-partitioning property guarantees that the rule with the maximum global evaluation will exceed the global best-so-far on some processor. Initially there will be a flurry of communication, until a rule is found with a large global evaluation. Communication will then occur only if a learner finds a rule that exceeds this threshold.

Another benefit of cooperation is that one learner can reduce its search based on knowledge learned by another learner. A thorough treatment of pruning for rule-learning search is beyond the scope of this paper, but Webb (1995) discusses how massive portions of the search space can be pruned in the admissible search for the rule that maximizes the Laplace accuracy estimate. In a distributed setting, if a learner discovers that a portion of the space is guaranteed not to contain satisfactory rules, it can share this knowledge with the other learners. Consider a simple, intuitive example: we are not interested in rules whose coverage is below a certain level. When a learner finds a rule whose coverage is below threshold, it sends the rule out for review. If the review verifies that the rule is indeed below threshold globally, then the learner shares the rule with the group. It is guaranteed that every specialization of this rule will also be below threshold, so the portion of the rule space below this rule can be pruned. Webb shows how the search space can be rearranged dynamically to maximize the effect of pruning.

Cooperation can also be used to reduce search heuristically. Rule-learning programs are used primarily for two types of learning: (i) discovery of rules that individually are interesting to domain experts, *e.g.*, in the fraud domain, and (ii) learning a disjunctive set of rules that are used to build a classifier, *e.g.*, a decision list (Clark & Niblett, 1989). Often the basis for building classifiers is the common "covering" heuristic: *iteratively learn rules that cover at least one example not covered by the current rule set* (Michalski, *et al.*, 1986; Clark & Niblett, 1989; Segal & Etzioni, 1994). Distributed learning systems can get much leverage from cooperation based on the covering heuristic. Specifically, as individual learners find good rules they can share them with the group. Allowing different learners to search the space in different orders will increase the effect of the cooperation. Consider the following extreme example: a large search space contains 10 rules that together cover the example set, and there are 10 distributed learners each of which starts its search with a different one of these rules. In this case, after each learner searches 1 rule (plus the subsequent review and sharing), the learning is complete. We hypothesize that such cooperation can lead to superlinear speedups over a monolithic learner (*cf.*, work on superlinear speedups for constraint satisfaction problems (Kornfeld, 1982; Clearwater, *et al.*, 1991)).

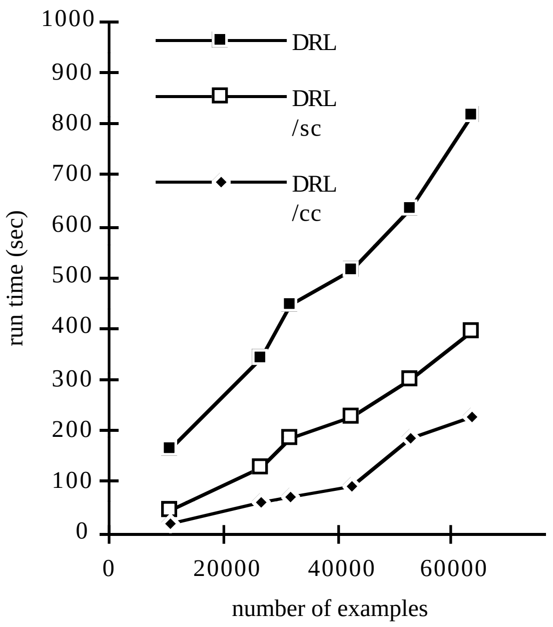

Figure 2. The effect of the covering heuristic on DRL using 2 workstations + 1 for rule review. /sc denotes simple covering. /cc denotes cooperative covering.

Figure 2 shows the effects of the covering heuristic on the run time of DRL (averages over 3 runs). Let us distinguish between *simple covering (/sc)*, using the covering heuristic within a run of RL to cover an example (sub)set, and *cooperative covering (/cc)*, sharing learned rules to cover the example set by the ensemble of learners. As shown in the figure, for these runs simple covering provided approximately a factor of two speedup over DRL without covering. Cooperative covering provided another factor of two speedup, on the average.

Related Work: Scaling Up Machine Learning

There are several approaches one might take to apply symbolic machine learning to very large problems. A straightforward, albeit limited, strategy for scaling up is to use a fast, simple method. Holte (1993) showed that degenerate decision trees, *decision stumps*, performed well for many commonly used databases. While the algorithm for learning decision stumps is fast, the method prohibits the learning of complex concepts.

A second strategy is to optimize a learning program's search and representation as much as possible, which may involve the identification of constraints that can be exploited to reduce algorithm complexity, or the use of more efficient data structures (Segal and Etzioni, 1994; Webb, 1995). These techniques are complementary to the scaling obtained by distributed processing.

The most common method for coping with the infeasibility of learning from very large data sets is to select a smaller sample from the initial data set. Catlett (1991b) studied a variety of strategies for sampling from a large data set. Despite the advantages of certain sampling strategies, Catlett concluded that they are not a solution to the problem of scaling up to very large data sets. Fayyad, *et al.* (1993), use sampling techniques, *inter alia*, to reduce a huge data set (over 3 terabytes of raw data). One method they use is to partition the data set, learn rules from subsamples, and use a covering algorithm to combine the rules. This method is similar to incremental batch learning and coarse-grained parallel methods (both described below). Catlett (1991b; 1992) also found that by looking at subsets when searching for good split values for numeric attributes, the run time of decision-tree learners can be reduced, without a corresponding loss in accuracy.

Incremental batch learning (Clearwater, *et al.*, 1989; Provost & Buchanan, 1995), a cross between sampling and incremental learning, processes subsamples of examples in sequence to learn from large training sets. Such an approach is effective for scaling up because even for learners that scale up linearly in the number of examples, if the example set does not fit in main memory, memory-management thrashing can render the learner useless. Such methods can take advantage of the invariant-partitioning property and the covering heuristic to approximate the effects of cooperation, as in a serial version of DRL.

Gaines (1989) analyzed the extent that prior knowledge reduces the amount of data needed for effective learning. Unfortunately, pinpointing a small set of relevant domain knowledge begs the very question of machine learning. Aronis and Provost (1994) use parallelism to enable the use of massive networks of domain knowledge to aid in constructing new terms for inductive learning.

Finally, three approaches to decomposition and parallelization can be identified. First, in *rule-space* parallelization, the search of the rule space is decomposed such that different processors search different portions of the rule space in parallel (Cook and Holder, 1990). However, this type of parallelization does not address the problem of scaling up to very large data sets.

The second parallelization approach, taken by Lathrop, *et al.* (1990), and by Provost and Aronis (1996), utilizes *parallel matching*, in which the example set is distributed to the processors of a massively parallel machine. Provost and Aronis show that the parallel-matching approach can scale a rule-learner up to millions of training data. Our work differs from the massively parallel approaches in that our goal is to take advantage of existing (and often under-utilized) networked workstations, rather than expensive parallel machines.

Finally, our work is best categorized by the third approach to parallel learning, the *coarse-grained* approach, in which the data are divided among a set of powerful processors. Each processor (in parallel) learns a concept description from its set of examples, and the concept descriptions are combined. Brazdil and Torgo (1990) take an approach similar to a distributed version of the approach of Fayyad, *et al.*, (described above), in which a covering algorithm is used to combine rules learned from the subsets, but they do not experiment with very large data sets. Chan and Stolfo (1993) also take a coarse-grained approach and allow different learning programs to run on different processors. Not unexpectedly, as with sampling, such techniques may degrade classification accuracy compared to learning with the entire data set. This degradation has been addressed by learning to combine evidence from the several learned concept descriptions (Chan & Stolfo, 1994). Our method differs from other coarse-grained parallel learners (and from incremental batch learning, and the method of Fayyad, *et al.*), because it utilizes cooperation between the distributed learners. Cooperation allows guarantees to be made about performance of learned rules relative to the entire data set, and can yield substantial speedups due to sharing of learned knowledge.

Conclusion

We demonstrate a powerful yet practical approach to the use of parallel processing for addressing the problem of machine learning on very large data sets. DRL does not require the use of expensive, highly specialized, massively parallel hardware. Rather, it takes advantage of more readily available, conventional hardware making it more broadly applicable. Furthermore, DRL provides a performance guarantee.

Preliminary results indicate that we can scale up by another order of magnitude by further optimizing the search of the underlying learning system. For the fraud data, a prototype system that uses spreading activation instead of matching as the basic learning operation learns from 100,000 examples plus hierarchical background knowledge in under 5 minutes and 1,000,000 examples in five hours (Aronis & Provost, 1996). This suggests that the DRL system (as configured above) using spreading activation instead of matching will learn from 1,000,000 examples in about an hour.

Acknowledgements

John Aronis has been involved in many stimulating discussions on scaling up machine learning. The NYNEX S & T Machine Learning Project and the University of Pittsburgh Dept. of Medical Informatics provided support.

References

Aronis, J. M., & Provost, F. J. (1994). Efficiently Constructing Relational Features from Background Knowledge for Inductive Machine Learning. In *Proceedings of the AAAI-94 Workshop on KDD.*

Aronis, J. & Provost, F. (1996). Using Spreading Activation for Increased Efficiency in Inductive Learning. Intelligent Systems Lab, Univ of Pittsburgh, Tech Report ISL-96-7.

Brazdil, P. & Torgo, L. (1990). Knowledge Acquisition via Knowledge Integration. In Wielinga (ed.), *Current Trends in Knowledge Acquisition,* 90-104. Amsterdam: IOS Press.

Buchanan, B., & Mitchell, T. (1978). Model-directed Learning of Production Rules. In Waterman & Hayes-Roth (ed.), *Pattern Directed Inference Systems*. Academic Press.

Catlett, J. (1991a). Megainduction: a Test Flight. In *Proceedings of the Eighth International Workshop on Machine Learning*, p. 596-599. Morgan Kaufmann.

Catlett, J. (1991b). *Megainduction: machine learning on very large databases*. Ph.D. Thesis, University of Technology, Sydney.

Catlett, J. (1992). Peepholing: choosing attributes efficiently for megainduction. In *Proceedings of the Ninth Int. Conf. on Machine Learning,* 49-54. Morgan Kaufmann.

Chan, P., & Stolfo, S. (1993). Toward Parallel and Distributed Learning by Meta-Learning. In *Proceedings of the AAAI-93 Workshop on KDD*.

Chan, P., & Stolfo, S. (1994). Toward Scalable and Parallel Inductive Learning: A Case Study in Splice Junction Prediction. In the working notes of the ML-94 Workshop on Machine Learning and Molecular Biology.

Clark, P., & Boswell, R. (1991). Rule Induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, p. 151-163.

Clark, P., & Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning*, **3**, p. 261-283.

Clearwater, S., Cheng, T., Hirsh, H., & Buchanan, B. (1989). Incremental batch learning. In *Proc. of the 6th Int. Wkshp on Machine Learning*, 366-370. Morgan Kaufmann.

Clearwater, S., Huberman, B., & Hogg, T. (1991). Cooperative Solution of Constraint Satisfaction Problems. *Science*, **254**(1991), p. 1181-1183.

Clearwater, S., & Provost, F. (1990). RL4: A Tool for Knowledge-Based Induction. In *Proc. of the 2nd Int. IEEE Conf. on Tools for AI*, p. 24-30. IEEE C.S. Press.

Cook, D., & Holder, L. (1990). Accelerated Learning on the Connection Machine. In *Proc. of the 2nd IEEE Symp. on Parallel and Distributed Processing*, p. 448-454.

Fayyad, U., Weir, N., & Djorgovski, S. (1993). SKICAT: A Machine Learning System for Automated Cataloging of Large Scale Sky Surveys. In *Proc. of the Tenth Int. Conf. on Machine Learning*, p. 112-119. Morgan Kaufmann.

Gaines, B. R. (1989). An Ounce of Knowledge is Worth a Ton of Data. In *Proc. of the Sixth Int. Workshop on Machine Learning*, p. 156-159. Morgan Kaufmann.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, **11**(1), p. 63-90.

Kornfeld, W. A. (1982). Combinatorially Implosive Algorithms. *Comm. of the ACM*, **25**(10), p. 734-738.

Lathrop, R. H., Webster, T. A., Smith, T. F., & Winston, P. H. (1990). ARIEL: A Massively Parallel Symbolic Learning Assistant for Protein Structure/Function. In *AI at MIT: Expanding Frontiers.* Cambridge, MA: MIT Press.

Michalski, R., Mozetic, I., Hong, J., & Lavrac, N. (1986). The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of AAAI-86*, p. 1041-1045. AAAI-Press.

Provost, F. J., & Aronis, J. (1996). Scaling Up Inductive Learning with Massive Parallelism. *Machine Learning*, **23**.

Provost, F. J., & Buchanan, B. G. (1995). Inductive Policy: The Pragmatics of Bias Selection. *Machine Learning*, **20**(1/2), p. 35-61.

Provost, F., & Hennessy, D. (1994). Distributed machine learning: scaling up with coarse-grained parallelism. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*.

Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, **1**, p. 81-106.

Quinlan, J. (1987). Generating production rules from decision trees. In *Proceedings of IJCAI-87*, p. 304-307. Morgan Kaufmann.

Quinlan, J. R., & Cameron-Jones, R. M. (1995). Oversearching and Layered Search in Empirical Learning. In *Proc. of IJCAI-95*, 1019-1024. Morgan Kaufmann.

Segal, R., & Etzioni, O. (1994). Learning Decision Lists using Homogeneous Rules. In *Proceedings of AAAI-94*, p. 619-625. AAAI Press.

Webb, G. I. (1995). OPUS: An Efficient Admissible Algorithm for Unordered Search. *Journal of Artificial Intelligence Research*, **3**, p. 431-465.

Weiss, S. M., Galen, R. S., & Tadepalli, P. V. (1990). Maximizing the Predictive Value of Production Rules. *Artificial Intelligence*, **45**, p. 47-71.